**The Ultimate JavaScript Full Stack roadmap for 2022**

When we start to learn software development, we feel overwhelmed with the amount of messy information that exists, we are not very clear about what to learn first, and many times we end up getting discouraged because we do not have a clear learning path, that’s why I want to share with you what I consider the learning path to be a Full Stack developer in 2022.

Let’s start with the basics, what every software developer or programmer must know

**1. Git, GitHub Workflow**

Learn Git. Git is the default version control system, with this tool you will be able to version the code base of any project, in addition to reverting the changes you have made whenever you want, without a doubt it is the first thing you should learn.

Then create an account on [GitHub](https://github.com/), it’s free, and it will be very useful for uploading projects from your computer to the cloud, which in git is known as a remote repository. You will be able to collaborate on other people’s projects and receive collaborations on your own, think of Facebook for developers, in addition, any modern company uses GitHub to manage its projects, so being on this social network and understanding it will be very useful, there are other alternatives like GitLab or Bit Bucket, but GitHub is the default standard.

**2. Terminal and command line basics**

If you want to dedicate yourself to computer science, you must necessarily know the basic concepts of the terminal and command line, since the tools that developers use in their day to day rarely have a graphical interface incorporated, in addition the terminal is much faster than any interface, because it interacts directly with operating system resources.

**3. Linux basics**

If you want to dedicate yourself to web development, learning Linux will not be completely mandatory, but if it is an important skill for any software developer or person who is dedicated to computing in general, knowing the basics is enough for now.

**4. Client/Server architecture**

You must know in detail the client-server architecture, how a request is made from the client to the server, how the server responds to the client, which is HTTP, because we interact with the database from the server and not in the client, and different important concepts.

**Frontend:**

**5. HTML**

If you want to dedicate yourself to web development, knowing HTML is practically mandatory, it is a HyperText Markup Language, with this language we give the structure to the web, imagine that the web is a person, HTML would be the skeleton. It is a relatively easy language, it will not take you more than 3 afternoons to understand it.

**6. CSS**

The reason websites are beautiful today is because of CSS, if HTML is the skeleton of the web, CSS is the clothes. With this language, we will give all the styles to our ugly and boring HTML. We will give styles to the boxes, buttons, forms, etc. With CSS, we will make our website look good on any device, be it a mobile, tablet or computer, thanks to the responsive design.

**7. JavaScript Fundamentals**

JavaScript is the default programming language of the web, if we want to become a web developer we must learn this language. Mastering JavaScript will take time, it is not something that can be achieved in a week, but you can learn little by little, the good thing is that you do not need to be an expert in JavaScript to make websites or applications.

**8. TypeScript (Basic to Advanced)**

Once we understand JavaScript and have done a few things with it. We must learn TypeScript, TypeScript is the solution to many of the problems of JavaScript, it is designed for the development of robust applications, implementing features in the language that allow us to develop more advanced tools for application development. In medium or large projects, and especially in a development team, writing code in TS offers great advantages that will be noticed in the short and long term.

**9. DOM Manipulation**

When writing web pages and applications, one of the most common things you’ll want to do is manipulate the structure of the document in some way. Typically, this is done using the Document Object Model (DOM), a set of APIs for controlling HTML and styling information that makes heavy use of the Document object.

**10. AJAX**

AJAX is a technique to communicate with the server without the need to reload the page, example: When you are on Instagram or any social network, and you want to load all the comments of a publication, the application shows them to you without the need to reload the page, this before from AJAX was impossible.

**11. Fetch API, Axios**

After understanding AJAX, you must learn how to request data from an external API or server, and this is achieved with the browser API called Fetch.  
This data comes in a data exchange format called JSON, you can practice bringing data from test APIs like [jsonplaceholder](https://jsonplaceholder.typicode.com/" \t "_blank), after understanding Fetch, you can use a library like Axios, a modern library widely used today to make calls to the server.

**12. SASS, LESS or Stylus (Choose one)**

Learning a CSS preprocessor will give us a significant advantage when writing our styles, I remember when I learned SASS, I didn’t want to go back to native CSS, it improved how fast I was writing CSS code significantly. You can learn SASS, LESS or Stylus, each one has its pros and interesting features, it doesn’t matter which one you learn, SASS is the most used, and it is the one that you will surely find in companies.

**13. React, Vue or Angular (Choose one)**

Learning a JavaScript framework will be complex at first, but you will notice that it changes dramatically over time. I could write 10 stories about the advantages of using a JavaScript framework over native or vanilla JavaScript, but the main advantage you will notice is that your code is much more understandable, more maintainable. All 3 are highly used, I would dare to say that React is the most popular and the one with the most job opportunities, but in the world of development everything changes quickly. Likewise, pick one and learn it, don’t get stuck on native JavaScript, there are very few job opportunities in native JavaScript.

**14. Global state management with Redux**

We already have the bases of our framework, now let’s learn Redux.

Redux is an excellent tool for managing the state of an application. Its main benefits are:

* Global and immutable state
* Greater control of application state and data flow
* Scalable data architecture

**15. Linters**

A linter is a tool that helps us to follow the good practices or style guides of our source code. Each linter has its own rules that you can use to structure the code to your liking and ensure that the entire application follows a defined pattern and good practices. There are many, the most popular for JavaScript is ESLint.

**16. Server side rendering framework (Next.JS for React — Nuxt.JS for Vue, or Angular Universal for Angular)**

The advantage that a Server Side Rendering Framework gives us is search engine optimization (SEO), when we work with React, Vue or Angular, our application runs only on the client side, affecting the positioning of our page in search engines.

**17. Deployment**

With all this knowledge, we should be able to create a simple website or application, but so that people can enter to see our app, we must upload it to a hosting, there are several platforms, paid and free, [Vercel](https://vercel.com/" \t "_blank) or [Netlify](https://netlify.com/) allow us to upload our application or website extremely quickly.

**18. Unit Testing**

An extremely important skill that we must develop as programmers is to be able to test our solutions, in JavaScript, the Standard for this is [Jest](https://jestjs.io/). Thanks to testing, when we want to make a change in our application, we make sure not to break anything, in addition to making sure we write quality, robust code.

**19. End-to-End Testing**

End-to-end testing verifies that all components of a system can run and perform optimally in real-world scenarios. End-to-end tests are usually heavier than unit tests, since they verify the correct operation of the entire system, unlike unit tests that only verify a single functionality separately.  
The Standard for end-to-end testing is undoubtedly Cypress, and it is the one I would recommend you learn.

**Backend:**

**20. Node.JS Fundamentals**

If we think of server-side JavaScript we think of Node.JS by default, other alternatives are coming out recently, but Node continues to dominate and is the most used today, so I would recommend learning it, also if you already know JavaScript, learn Node It will be very easy for you.

**21. HTTP Methods**

Something fundamental when working with servers is to understand HTTP, what it is, and the different methods that exist, what each one is for, you should understand this before starting to create your first REST API.

**22. Express (Routes, Controllers, Errors handling…)**

The most used framework without a doubt and the default standard in Node is Express, this will bring us all the benefits of a framework applied to the Backend. Learn how routes work, what is a controller, a service, error handling.

**23. Environment Variables**

When we work with other developers or publish our code to GitHub, the code becomes public, and generally we will have to work with secret credentials that we do not want anyone to see, such as Authentication data to a service, passwords, secret keys, etc. That’s what environment variables are for, they are variables that change depending on the environment where our application is running and are not uploaded to the GitHub repository.

**24. Express Middlewares**

Imagine that we have a course application, but we want to validate that only users with the Teacher Role can upload courses and not everyone, for that we must intercept each course upload request, and verify if the user has the role of Professor, this is achieved with middleware, in express it is easy to achieve this, throughout an application we will need to create several middleware according to the logic of our business.

**25. Authentication with JSON Web Tokens**

Something fundamental in any application is managing user authentication, to recognize each user in our system and verify that they meet the requirements to access the resources of our application.

There are several techniques to achieve this, one of the most used are the JWT or JSON Web Tokens, they are unique tokens that identify each user and contain relevant information such as the id, in this way when we want to protect a resource in our application, we will verify the JWT that comes to us in the request, and we decide whether or not the user has access to the protected resources, according to our business logic, this can be, for example, access to Premium resources.

**26. MongoDB with mongoose ORM**

Any application needs a database to store user information, this is something basic, there are two different types of databases, relational databases, which use the SQL language, and non-relational databases that structure the data in objects similar to JSON, when I started with the backend, I became more familiar with non-relational databases, since they look like JavaScript objects, I recommend that you learn MongoDB, it is an extremely popular database, widely used today in day, but using MongoDB in Node.js can be a bit complex, for that there are ORMs, they greatly simplify the work of connecting with databases, for MongoDB the default standard is mongoose.

**27. SQL Basics**

But you shouldn’t just stick with NoSQL, learning SQL will make you a more complete programmer, I recommend learning the basics, then you can go deeper.

**28. TypeORM**

You can work with native SQL in your applications if you want, but it’s complex, ORMs make your life a lot easier when you work with databases, for relational databases I recommend learning TypeORM, it’s relatively simple, plus one of the great The advantages that an ORM gives us is the possibility of changing the database engine with ease, we can migrate our application from MySQL to Postgres or from Postgres to Oracle with just a few lines of code.

**29. Password hashing**

When we work on an application, we must not save the passwords in plain text, this is a very serious security error, for that we must encrypt the password before saving them in the database. We achieve this with libraries such as bcrypt, one of the most used. Learning to encrypt passwords is undoubtedly one of the fundamental skills of a Backend Developer.

**30. Nest.JS**

When we work with small projects, pure express is enough for us, but in larger projects, the software architecture becomes vital, and a good architecture is practically mandatory if we don’t want to rack our brains with each small integration or new change. Nest.JS is an Express framework that gives us a modular, scalable architecture, provides us with a command line that saves us a lot of repetitive work and increases our productivity. I highly recommend that you learn it, plus it is becoming highly sought after by companies.

**31. Caching**

Cache is an extremely important technique in any application. It allows us to store any data in memory for subsequent requests, avoiding the overload of our server and database, as well as significantly reducing the response time of our server. I definitely recommend that you learn Cache, the most widely used cached database today is Redis.

**32. Deployment**

With all this, we should be able to create a REST API that connects to a database, performs read and write operations, and returns the data to the client. But for it to be online and anyone can access it, we need to upload it to a hosting service, the most used in Node is Heroku, it’s free, I recommend that you learn it and upload your applications there.

**Some extra concepts you can learn**

- CI/CD  
- Docker  
- AWS Services like S3, EC2  
- Kubernetes  
- DDD  
- TDD  
- Hexagonal Architecture

**Conclusion:**

Becoming a Full Stack developer is not an easy path, but with perseverance and discipline you can achieve it, the secret is to enjoy learning a little every day, and to practice what you learn a lot. I hope that with this roadmap you have a little more clarity on the path and do not get lost with so much noise on the internet on this subject.